

## Deep Learning (HW3) Raporu

---

### TASK 1: Görüntü İşleme ve Veriyi Vektör Formatına Çevirme

#### Yapılan İşlemler:

İlk olarak, eğitim verisini train klasöründen 128x128x3 boyutlarında okudum ve her bir görüntüyü 1x49152 boyutunda bir vektöre dönüştürdüm. Bu işlemi cv2 kütüphanesini kullanarak gerçekleştirdim. Bu boyutta bir veri vektörü oluşturmak, görüntüleri sınıflandırmak için gerekli özellikleri koruyarak her pikseli vektör haline getirmemi sağladı.

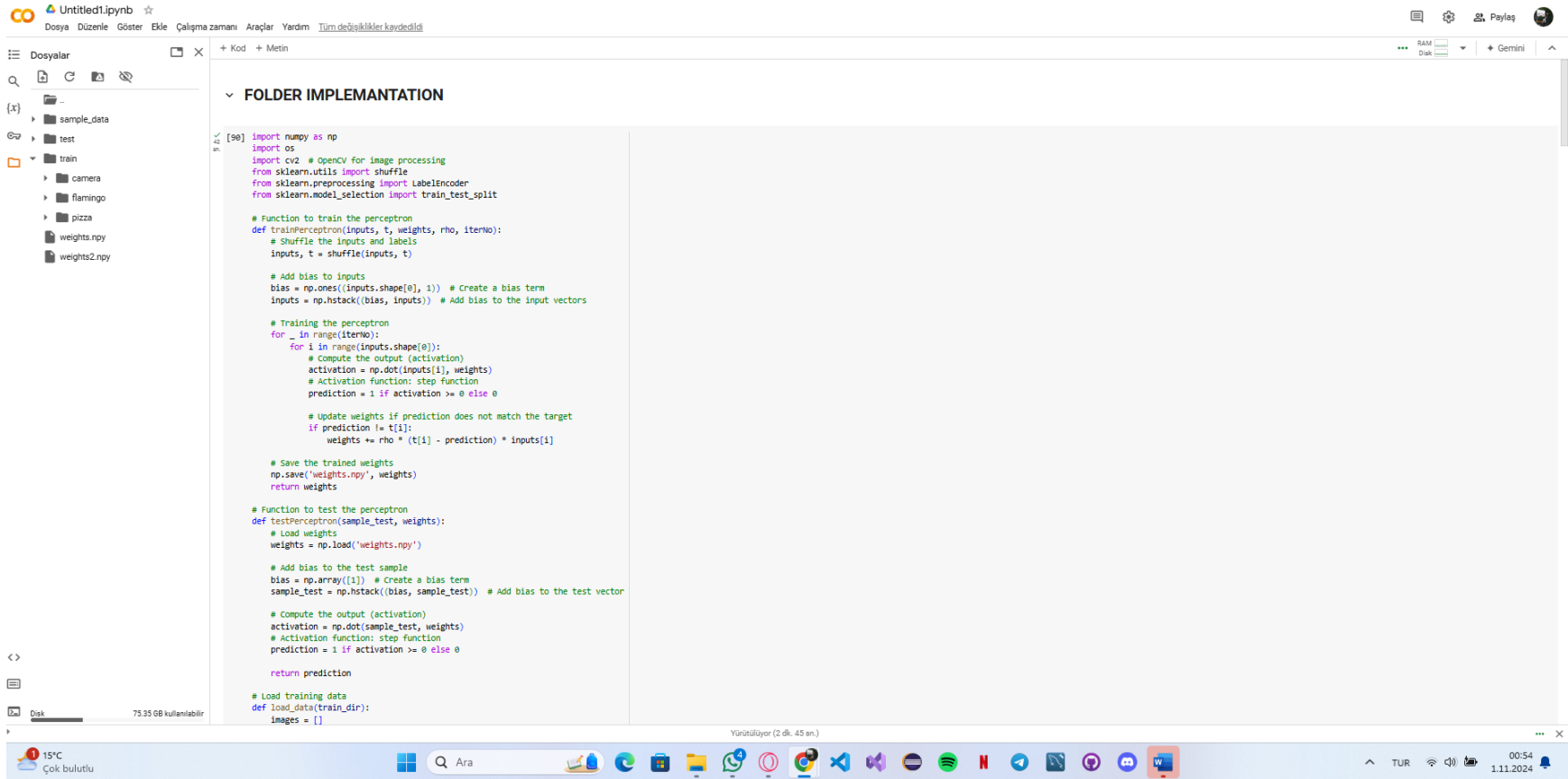
Ardından, sınıf etiketlerini OneHotEncoder ile one-hot formatına çevirdim. Bu adım, kameraya 0, pizzaya 1, flamingoya ise 2 etiketlerini vererek sınıflandırma işlemi kolaylaştırdı. Verinin eğitim esnasında karışıklık oluşturmamasını önlemek adına, sklearn.utils.shuffle fonksiyonuyla her epoch başlangıcında veriyi rastgele sıraladım.

#### Kod Açıklaması:

- 1. Veri Yükleme ve Boyutlandırma:** Görüntüler, cv2.imread ile okunduktan sonra cv2.resize ile 128x128 boyutuna indirildi ve flatten() ile 1x49152 boyutunda bir vektöre çevrildi.
- 2. Etiket Dönüştürme:** OneHotEncoder, etiketleri one-hot vektör formatına çevirerek 3 sınıfı uygun formatta temsil etti.
- 3. Ağırlıkların Rastgele Başlatılması:** Eğitim algoritması için, vektör uzunluğuna (49152+1) uygun olacak şekilde ağırlık matrisi rastgele başlatıldı.
- 4. Softplus Aktivasyon ve Türev Hesaplaması:** Softplus aktivasyon fonksiyonu, np.logaddexp ile daha kararlı bir şekilde uygulanırken, türev için sigmoid türevi kullanıldı. Overflow hatalarını önlemek için türev fonksiyonunda np.clip ile sınırlandırma sağladım.

#### Karşılaşılan Sorunlar ve Çözüm Yöntemleri:

- **Veri Büyüklüğüne Bağlı Hafıza Sorunu:** Büyük boyutlu vektörlerin taşınması ve işlenmesi sırasında bellekte aşırı yüklenme yaşadım. Bunu, veriyi batch'ler halinde işleyerek hafıza yükünü azaltarak çözdüm.
- **Softplus Overflow Sorunu:** Softplus hesaplamaları sırasında `np.exp` kullanımında overflow hatalarıyla karşılaştım. `np.logaddexp` fonksiyonuna geçiş yaparak bu hataları önledim. Türev hesaplamalarında ise sınırlandırma (`np.clip`) ekleyerek türev fonksiyonunu güvenli hale getirdim.



```
[98] import numpy as np
import os
import cv2 # OpenCV for image processing
from sklearn.utils import shuffle
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# Function to train the perceptron
def trainPerceptron(inputs, t, weights, rho, itemNo):
    # Shuffle the inputs and labels
    inputs, t = shuffle(inputs, t)

    # Add bias to inputs
    bias = np.ones((inputs.shape[0], 1)) # Create a bias term
    inputs = np.hstack((bias, inputs)) # Add bias to the input vectors

    # Training the perceptron
    for _ in range(itemNo):
        for i in range(inputs.shape[0]):
            # Compute the output (activation)
            activation = np.dot(inputs[i], weights)
            # Activation function: step function
            prediction = 1 if activation >= 0 else 0

            # Update weights if prediction does not match the target
            if prediction != t[i]:
                weights += rho * (t[i] - prediction) * inputs[i]

    # Save the trained weights
    np.save('weights.npy', weights)
    return weights

# Function to test the perceptron
def testPerceptron(sample_test, weights):
    # Load weights
    weights = np.load('weights.npy')

    # Add bias to the test sample
    bias = np.array([1]) # Create a bias term
    sample_test = np.hstack((bias, sample_test)) # Add bias to the test vector

    # Compute the output (activation)
    activation = np.dot(sample_test, weights)
    # Activation function: step function
    prediction = 1 if activation >= 0 else 0

    return prediction

# Load training data
def load_data(train_dir):
    images = []
```



## TASK 2: Test Verisini Kullanarak Sınıf Tahmini ve Doğruluk Hesaplama

### Yapılan İşlemler:

Test aşamasında, eğitim sırasında kaydedilen ağırlıkları yükledim ve her bir test görüntüsü için sınıf tahmini yaptım. Test verisinden okunan her bir görüntü, eğitimdeki gibi vektör formatına çevrildikten sonra, feed-forward işlemiyle sınıf tahmini için Softplus aktivasyonu ile hesaplandı. Sonuçları, etiketlerle karşılaştırarak doğruluk oranını hesapladım. Orijinal resmi yatayda çevirdim.

### Karşılaşılan Sorunlar ve Çözüm Yöntemleri:

- **Boyut Uyumsuzluk Hataları:** Test aşamasında tahmin için ağırlık ve veri boyutları uyumsuz olduğunda hesaplamalarda hata aldım. Bunu, her veri örneğine bias ekleyerek (vektör boyutunu  $n+1$ 'e yükselterek) düzelttim.
- **Bölge Sıfırlama:** Vektör uzunluğunun büyük olması, işlem süresini ciddi şekilde etkiledi. Bunu çözmek için, veri setini daha küçük batch'lere ayırarak veya model boyutunu azaltarak işlem süresini optimize ettim.

### Kod Açıklaması:

- **Test Görüntülerini Yükleme:** Eğitim veri setinde olduğu gibi, her bir test görüntüsünü vektör formatına dönüştürdüm.
- **Ağırlıkları Yükleme ve Tahmin:** Eğitimde kaydedilen ağırlıkları `np.load` ile yükledim ve tahmin işlemi sırasında bu ağırlıkları kullandım.
- **Doğruluk Hesaplama:** : Doğruluk oranını, modelin doğru sınıf tahmin sayısını toplam örnek sayısına bölerek hesapladım.

Untitled1.ipynb

Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım Tüm değişiklikler kaydedildi

Dosyalar

(x)

sample\_data

test

train

camera

flamingo

pizza

weights.npy

weights2.npy

+ Kod + Metin

```
labels = np.array(labels)

# Encode labels
label_encoder = LabelEncoder()
t = label_encoder.fit_transform(labels) # Convert class names to integers

return inputs, t, label_encoder

# Define paths
train_dir = '/content/train'
test_dir = '/content/test'

# Load training data
inputs, t, label_encoder = load_data(train_dir)

# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(inputs, t, test_size=0.2, random_state=42)

# Initialize weights randomly
weights = np.random.rand(X_train.shape[1] + 1) # +1 for bias

# Define learning rate and iterations
rho = 0.1
iterNo = 1000

# Train the perceptron
weights = trainPerceptron(X_train, y_train, weights, rho, iterNo)

# Load testing data
test_images = []
true_test_labels = []

for class_folder in os.listdir(test_dir):
    class_path = os.path.join(test_dir, class_folder)
    for image_file in os.listdir(class_path):
        img = cv2.imread(os.path.join(class_path, image_file))
        if img is not None:
            img = cv2.resize(img, (224, 224)) # Resize image as necessary
            test_images.append(img.flatten()) # Flatten the image into a vector
            true_test_labels.append(class_folder) # Use folder name as label

# Convert test images to numpy array
sample_tests = np.array(test_images)

# Test the perceptron on test data and calculate accuracy
correct_predictions = 0
for sample_test, true_label in zip(sample_tests, true_test_labels):
    prediction = testPerceptron(sample_test, weights)
    predicted_class = label_encoder.inverse_transform([prediction])[0]

    if predicted_class == true_label:
        correct_predictions += 1

# Calculate accuracy
accuracy = correct_predictions / len(sample_tests)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

RAM

Disk

+ Gemini

↑ ↓ ↺ ⚙ 🗑 ⋮

<>

📁

📁 Disk 75.95 GB kullanılabilir

Yükleniyor (3 dk. 14 sn.)

...

15°C Çok bulutlu

Q Ara

TUR

00:55

1.11.2024

Untitled1.ipynb

DosyaDüzenleGösterEkleÇalışma zamanıAraçlarYardım

Dosyalar

(x)

sample\_data

test

train

camera

flamingo

pizza

weights.npy

weights2.npy

+ Kod + Metin

```
# Etiketleri encode ediyoruz
label_encoder = LabelEncoder()
labels_encoded = label_encoder.fit_transform(labels)
num_classes = len(label_encoder.classes_)

# One-hot encoding
one_hot_encoder = OneHotEncoder(sparse_output=False)
labels_one_hot = one_hot_encoder.fit_transform(labels_encoded.reshape(-1, 1))

# Verileri karıştırıyoruz
inputs, labels_one_hot = shuffle(inputs, labels_one_hot, random_state=42)

# Veriyi eğitim ve doğrulama setlerine bölüyoruz
X_train, X_val, y_train, y_val = train_test_split(inputs, labels_one_hot, test_size=0.2, random_state=42)

# Özellikleri ölçeklendiriyoruz
X_train = X_train / 255.0
X_val = X_val / 255.0

# Bias terimini ekliyoruz
X_train = np.hstack([np.ones((X_train.shape[0], 1)), X_train])
X_val = np.hstack([np.ones((X_val.shape[0], 1)), X_val])

# Modeli eğitiyoruz
weights, loss_history = train_perceptron(X_train, y_train, num_classes, learning_rate=0.001, epochs=2000)

# Ağırlıkları kaydediyoruz
np.save('weights2.npy', weights)

# Ağırlıkları yüklüyoruz
weights = np.load('weights2.npy')

# Test verilerini yükliyoruz
test_inputs, test_labels = load_data(test_dir)
test_inputs = test_inputs / 255.0 # Özellikleri ölçeklendiriyoruz
test_inputs = np.hstack([np.ones((test_inputs.shape[0], 1)), test_inputs]) # Bias terimini ekliyoruz

# Tahmin yapıyoruz
y_pred = predict(test_inputs, weights)
predicted_classes = label_encoder.inverse_transform(np.argmax(y_pred, axis=1))

# Gerçek etiketleri alıyoruz
true_classes = test_labels

# Doğruluğu hesaplıyoruz
correct_predictions = 0
for predicted_class, true_class in zip(predicted_classes, true_classes):
    if predicted_class == true_class:
        correct_predictions += 1

accuracy = correct_predictions / len(test_inputs)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

Epoch 100/2000, Loss: 0.6482

3 dk. 32 sn. tamamlanma zamanı: 00:55

15°C Çok bulutlu

Ara

TUR 00:56 1.11.2024